VZDU # 29 MARCH/APRIL 1991.

VZ DOWN UNDER CLUB

Editor	Treasurer
Harry Huggins	Ron Allen
12 Thomas Str.	2 Orlando str.
Mitcham. 3132.	Hampton.3188.
03-873 1408	03-598 4534

I am now on the mend, so should be able to catch up on the backlog I have here.

I am still not getting feedback from our members, so I assume that what I'm doing is alright. Even the last Scream Sheet did'nt throw any brick-bats. (No Bouquets either.)

You will notice I have added an extra column to the High Scores. You may use this for degree of difficulty or for Joysticks or keyboard.

We have in this issue another very good article from Bob Kitch. (He did'nt go with the floods after all). This will be of special interest to newcomers to the VZ world. It is a history of the VZ clubs.

Also the third of the series on Sound for the VZ., also by Bob. And the continuing series by David Wood on Adventure Program Writing. I have on hand his next 3 articles of the series, so there is a lot in them. He has also sent a demonstration program. It is too long to print in the Newsletter, but those that would like it can borrow it the same as back issues. Even if you don't want to write Adventure Programs, the series is well worth following, as it covers a lot of general basic programming, and shows the power of the VZ.

I'll make a suggestion. Some time ago we sent a letter to ALL Dick Smith stores, advising them of our activities. I have heard that some are not sure we are still going. So, whenever in a DSE store, remind them we are Still going strong! And if you write to DSE head office, suggest they make mention in their circulars to their stores and dealers. They can at least do that for the VZ of which they sold thousands.

Cheerio for this issue, Harry

A HISTORY OF VZ USER GROUPS IN AUSTRALIA AND NEW ZEALAND.

by Bob Kitch.

The first advertisement for the sale of DSE's VZ-200 computer was in Electronics Australia June 1983. That is a long while ago, particularly in computing terms, for a small and cheap 8-bit computer to survive. Since that time many people have purchased and used the VZ-200, and its upgrade in July 1985, the VZ-300. Users and owners of the VZ naturally tended to band together, to chew over mutual interests and problems, in much the same way as owners of other "breeds" of computers. These "jam" sessions were most often held over the phone, but have you ever tried to satisfactorily discuss a software problem over the phone? The next stage was to organize a meeting of interested enthusiasts, usually on a week-end, in someones home or at a conveniently located hall. And so began "A VZ USER GROUP".

The VZ was greatly assisted by its origins. The VZ was really a souped-up Tandy TRS-80 or its clone, the DSE System-80. These two machines were based on a Zilog Z-80 microprocessor and held a version of Microsoft BASIC in ROM. (this is what gives a microcomputer its distinctive personality) These machines were responsible for the commencement of the home and personal computing boom - no small claim. In 1983, the VZ-200, manufactured by Video Technology in Hong Kong, used an improved version of the Microsoft ROM, offered colour and sound, increased memory capacity and a low price. A lot of TRS-80 and Sytem-80 owners upgraded to the VZ. These guys often knew Z-80 Assembler and the workings of the Level II ROM backwards! They also bought a useful software base to the VZ.

The VZ computer quickly gained a large following and was clearly a marketting success for DSE. They claim to have sold in excess of 30,000 VZ-200. DSE's support for the VZ was often found wanting - a very common moan amongst Users. The various User Groups that sprang into existence provided the essential support for the VZ. Without them, the VZ and its Users would have probably withered away and gone the way a number of other small computers did.

User Groups usually have a small band of committed enthusiasts, who tend to carry the activities of the Group. They provide the core of knowledge that, unites the group, fascinates and nurtures the uninitiated to computing and possess a restlessness to achieve more with the VZ. Their reward seems to be simply seeing a new member become a successful User. A fairly natural activity to flow from this Group, is the production of a newsletter, to serve far-flung enthusiasts (remember the telephone) and to record discoveries (be they hardware tricks or new programs) for other Users.

A newsletter needs an Editor. These fellows are the greatest. Their contribution to VZ computing and the Users is huge. The amount of information recorded in User Groups Newsletters is staggering. Some of you may be aware of the magazine article, books and software lists that are available. A similar list of User Groups and their Newsletters is in preparation. Your assistance in adding to it would be appreciated.

With this background then, I provide a brief discussion of the VZ User Group Newsletters with which I am familiar. I apologize for any omissions, but would appreciate being made aware of any shortcomings.

1. DSE EFFORTS.

Undoubtedly, DSE's was aware of its poor support for the VZ - we told them often enough! They had also engaged the prolific Tim Hartnell to write a series of books for the VZ. Tim commenced the first VZ-200 User Group in mid-1983. He produced 3 editions of "VZ-200 Interface" - the Official Magazine of the VZ200 Users' Club. The last edition (#3) was issued around Easter 1984. Each edition was around 8 pages and filled with hints, Basic programs and DSE advertisements.

(As a sad aside, at the time of writing - February 1991 - I noticed in a recent paper that Tim Hartnell had succumbed to cancer and passed away aged 40. Tim was a significant contributor to the VZ with his VZ-200 and VZ-300 books.)

By mid-1984 other "private" User Groups had begun to form. DSE's dropped "Interface" and began producing "Comput", a newsletter covering other machines sold by DSE. Five editions wrer erratically produced from August 1984 to July 1986. Minor articles on the VZ are included.

I should mention here, forz completeness, that DSE's Annual Catalogue since 1983 has contained VZ information. The release of software and hardware items can be tracked through these.

2. THE FIRST USER GROUP CLUB.

Who said that Victorians couldn't read? In early 1984, Mr. Luigi Chiodo produced the first edition of "Output". He subsequently changed the name to "Visual Display Unit". I do not have a full set of his newsletters. I have #1 - #7. How many did Luigi produce? They contain a considerable amount of DSE supplied material as well as some other interesting contributions.

3. THE FIRST QUEENSLAND CLUB.

The first "private" VZ users group newsletter was produced by Mr. John D'Alton in June 1984. John went on to produce 27 issues of "LE'VZ News" with the last in May 1990. A monumental effort by John, particularly when considered with his software support, program writing and Christmas Meetings. In addition to the newsletter, John produced a book on programming hints and hardware. This is a most valuable and well produced set of newsletters and most are still available from John as back issues.

4. A GROUP IN ADELAIDE.

In July 1984, Mr. John Waters of Cheltenham in Adelaide produced the first edition of "Ve Zee News". John (perhaps wisely!) initially said that he would only produce 12 newsletters. True to his word, he produced 12 monthly editions from July 1984 to June 1985. His newsletters contain a lot of "meaty" information on the VZ. I suspect that a number of the contributors were ex-TRS-80 men. "Ve Zee News" is an excellent collection of newsletters and contains hardware modifications and software listings.

5. ACROSS THE TASMAN.

The VZ was sold by DSE stores in New Zealand and in July 1984 a user group commenced in Christchurch. They produced a "Christchurch VZ User Group Newsletter" on a monthly basis up until April 1988 when interest waned. Their newsletters are full of Basic programs and are very chatty with details of their regular monthly meetings.

6. LYSCO FROM WA.

6. LYSCO FROM WA.

The Leon Young Software Company sprang up in Perth in late 1984. (Actual dates are uncertain as Leon did not date his newsletters.) He also supported the Amstrad and Commodore computers. Leon's Newsletters and Catalogs were fairly informal notes but contained a number of useful tips on the VZ as well as advertisements for his software. I have about 10 newsletters in all dated around November 1984 to his wind-up in July 1986.

7. ANOTHER IN NZ.

In December 1984 another user group was underway in Auckland, New Zealand. They produced "XILOG The Microcomputer Magazine for VZ200, VZ300 and Aquarius Users" - suggesting their ties with DSE. I have an incomplete set of XILOG but it is full of interesting snippets for the VZ user. My last edition is #8 dated August 1985. Does anyone know any more about this club?

8. OUT WEST IN NSW.

Also in late 1984, Mr. Rick Swancott organized a small user group in the western suburbs of Sydney. They produced 2 small newsletters around December 1984, entitled "Out West VZ-200 User Group". It was filled with Basic games.

9. FROM THE TROPICS..

I first corresponded with Mr. Gordon Browell in Darwin in late 1985. Gordon was running the "Ad Lib VeeZee Micro Club". He produced informal notes and programs and freely circulated these to any interested correspondents. The informality of this arrangement subsequently resulted in his production of two excellent beginners series. The first was "Micro Magic - Beginners Guide to the Vee Zed" in 6 parts, and the second was "Studio Ad Lib - Micro Magic Workshop" also in 6 parts. They are excellently written and produced series.

Somewhere around 1985, Gordon moved to Biggenden in south-east Queensland but unfortunately he has more recently become involved in other computers.

10. ANOTHER BRISBANE GROUP.

Mr. Michael Novakovic, a secondary student at the time, and living at Goodna, A western suburb of Brisbane, produced 4 newsletters. It was called "VZCOMPU200/300" and ran from December 1985 to April 1986. Michael experimented with machine code and POKE's to the communication region!

11. YET ANOTHER IN NSW.

In January 1986, Mr. Mark Harwood published the first edition of "VZ User". He produced 22 issues and finished in September 1988. Mark was a Tertiary student undertaking Electrical Engineering studies. His newsletter contained a number of in-depth features on the VZ. He also gave an introduction to VZDOS-in-ROM that has not been bettered elsewhere. Mark also developed some excellent software in conjunction with Gavin Williamson of Laserlink.

12. ANOTHER NSW CLUB.

The Hunter Valley Region of NSW has been well served since June 1986 by the

"Hunter Valley VZ Users Group Newsletter" (now Journal). Mr Gavin Williamson produced the first 6 newsletters but Mr. Joe Leon has produced up to the current edition #32. This newsletter is still underway and is produced bi-monthly. The newsletters are consistently packed with programs and hardware modifications.

13. AND BACK TO VICTORIA.

Scott Le Brun was a very prolific writer of adventure games for the VZ. In August 1986 he began publication of "VeeZed Down Under". Scott also ran monthly meetings from his home. Scott produced 13 editions before the illustrious Harry Huggins took over in September 1988. Harry has since produced number 28 and is still going strong. Harry has also acquired the remaining VZ hardware from DSE. If you are ever passing through Melbourne, plan to spend an evening with Harry.

14. ANOTHER AUCKLAND CLUB.

In October 1986 Peter Hill commenced the Auckland VZ300/200 Users Club. Since that time peter has produced on a monthly basis a creditable 50 editions of "VZ-Link". The Auckland Club is still going strong and has a monthly meeting in the city - afternoon tea and a cuppa provided.

15. AN INFORMAL BRISBANE GROUP.

A group of enthusiasts in Brisbane decided to hold monthly meetings without producing a newsletter as, at the time, John D'Alton was carrying out this function. This group meets on the first Saturday of each month at Stan Noble's house. This group is evolving towards a computer interest group as most of the members have now purchased PC's. The VZ still gets a hammering and some new-comers regularly appear.

16. A NEW INNOVATION.

Last year Jason Oakley of Sapphire Productions produced 2 editions of his excellent "DiskMag". This is a magazine on disk for the VZ. Jason needs all the support he can get to continue his efforts.

TONE GENERATOR. *Cont. from page 10*

```
700 '***OFFSETS FOR PARAMETER STORAGE ABOVE ROUTINE.
710 DATA 81,82   :'ON INTERVAL   - PA%(0)
720 DATA 83,84   :'OFF INTERVAL  - PA%(1)
730 DATA 85,86   :'ON INCREMENT  - PA%(2)
740 DATA 87,88   :'OFF INCREMENT - PA%(3)
750 DATA 89,90   :'TONE DURATION - PA%(4)
759 '
760 '***OFFSETS FOR PARAMETERS IN MACHINE CODE ROUTINE.
770 DATA 3,4     :'ONINC
780 DATA 7,8     :'OFINC
790 DATA 11,12   :'TONDUR
800 DATA 17,18   :'ONDUR
810 DATA 21,22   :'ONDUR
820 DATA 34,35   :'OFDUR
830 DATA 38,39   :'OFDUR
840 STOP
999 '
```



```
10000 '***UPDATE DISK FILE.
10010 CLS:PRINT@200,"ERASING FILE":ERA"TONEGEN"
10020 PRINT@200,"SAVING FILE ":SAVE"TONEGEN"
10030 STOP
19999 '
20000 '***PRINTER DUMP FOR DEBUGGING.
20001 '***TO ACTIVATE TAKE OUT REMARK IN LINE #670.
20010 LPRINT"DEBUG OUTPUT FROM TONEGEN":LPRINT
20020 LPRINT"TOM =",TM:LPRINT
20030 LPRINT"USR() ADDR. =",PEEK(30862),PEEK(30863):LPRINT
20040 LPRINT"PARAMETERS"
20050 FOR I%=0 TO 4
20060 LPRINT MS$(I%),I%,PA%(I%)
20070 NEXT I%:LPRINT
20080 LPRINT"POINTERS IN VA%()"
20090 FOR I%=0 TO 23
20100 LPRINTI%,VA%(0,I%),VA%(1,I%),VA%(2,I%)
20110 NEXT I%
20120 END
```

GAMES COLUMN

BY Paul Frantz

Back with a vengence! The VZDU games column has had an influx of contributions! Thanks to all those who put in the effort. I hope others will see the light and take the time to write in. At the moment I am recovering from the mountain of correspondence and so if you have sent an envelope for a personal reply, please bear with me.

One suggestion I received was from Mr Colin Hermans who recommends the following material (as well as David's game writing column) as well worth reading for info regarding writing adventure games. In VADU#3 (Nov 1986) and VZDU#5 (Mar/April 1987) there is an article written by Scott Le Brun. There also is a section in 'THE GIANT BOOK OF GAMES FOR THE VZ300' written by Tim Hartnell which may be of use. This book was published by D.S.E. so you may be lucky to still find it. Thanks for the suggestions Colin! If anyone else has read some useful information on the subject, please aleet me to it at the address below!

And now to the review! Actually I am starting to run out of games to review as of late. Maybe next time I'll have a whole new bunch to review as a result of David's series! We wait in hope. Anyway onward to rescue .....

THE ENCHANTED PRINCESS

The story is simple but heartbreaking. One terrible day an evil wizard appeared for just long enough to grab the fair princess and carry her away to a far and distant place. Do you dare to take up the challenge of rescuing the princess and risking being changed into a variety of animals? The decision is yours (although copping out isn't nearly as much fun).

ENCHANTED PRINCESS was written by Gary McCleary who has added a very nice graphical touch at the start (you'll have to play it to see it). The style of the game is fairly standard though which is alright as its not the newest of games. What it does do however is constantly keep you informed with all the contents of the room displayed on the screen which can be a good thing for beginners. It also lists all the rooms' exits as well which can be handy. The big gripes over this one though are the lack of verbs and the irregular movement command. As all budding computer adventurers would know, a simple "N","S","E" or "W" is standard for ease of movement, well ENCHANTED PRINCESS dares to be different by making us enter such disasterously long commands like "GO E". This is just not on! Still it makes for a good game for beginners although it may frustrate some over the lack of a variety of verbs.

RATING:

*VOCABULARY:	4
*PUZZLE DIFFICULTY:	7
*ATMOSPHERE:	7
*LASTABILITY:	6
OVERALL........................	6

AVAILIBILITY: Not sure. Maybe Harry can help out here?

HOT, HANDY & HELPFUL HINTS

*Again no hints this month. It seems everyone's turned their attention to .....

QUESTIONS & QUERIES

? Ian Niedzwiecki writes: In HAUNTED MANSION 1. Where is the cross? 2. How do you kill the ghost?

? From Peter Watson.....Is PHAROAH'S CURSE the same as PHAROAH'S TOMB? ANSWER: No. PHAROAH'S CURSE is an arcade type game by D.S.E. whilst PHAROAH'S TOMB (reviewed last time) is an adventure game from LYSCO (I think).

? Does anyone know how to get out of the forest in MAGNUM QUEST? (asks Ben Hobson) I have tried move, go northe etc.. & also dig but I am stuck. I have taken a different combination of tools etc. but nothing.

? And finally from Bernice O'Mahoney: Our GALAXON game only allows a high score of 5 digits, so making the highest possible score 99999. Are there two games or is the high score of 150000 a misprint? ANSWER: 150000 isn't a misprint. When the score clock ticks over 99999, a small graphical mess comes up where the one hundred thousand digit should be and the rest of the score ticks on thus you can work out your score as the game doesn't stop.

Well we are nearing the end of yet another column. Below should be the new updated high score list and the competitiion is fast becoming a force to be reckoned with. Again for those who have written in, you should be hearing from me soon.

Well as always, see you next edition!

Please send all your hints, high scores, questions and answers in to:

Paul Frantz
25 Crocker St
KIRWAN QLD.4817

CORRECTION

Ben advises that he left these lines out of Death Maze.

```
50 V=3:JY=0:W=1:S=0:HS=0: GOSUB1180:GOTO1280
106 FOR Z=1TO20:NEXT Z:NEXT B
1281 IF S<OTHEN S=0
1282 IF S> HS THEN HS=S
1283 S$=RIGHT$("00000"+ RIGHT$(STR$(S), LEN(STR$(S))-1),5)
1284 HS$= RIGHT$("00000"+ RIGHT$(STR$(HS), LEN(STR$(HS))-1),5)
1332 PRINT@403,S$:PRINT@435,HS$
```

Peter Watson says he gets a NEXT without FOR error in 106. It does appear there is a surplus NEXT in 105-106. (Ed)

LET'S INVESTIGATE SOUND ON THE V.Z.

By Bob Kitch

Part III

In the last two articles, we played around with the SOUND command in the BASIC Interpeter and also got into the BEEP routine (using the USR command) in the VZeds ROM. I have also provided a brief outline of the manner in which sound is generated, the component parts of a sound wave-form and the limitations on sound generation on the VZ. In this article I will present a "full-blown" psuedo-assembler routine to fully explore the sound generating capabilities of the VZ. The I/O latch at 6800H is directly controlled - the sound routines in ROM are not used. The machine code is called up by a BASIC program that also passes the necessary parameters to the low-level program. The TONEGEN program is a good example of linking machine code to a BASIC program - a technique that I call FAST BASIC. It is an exceptionally powerful programming technique to get the utmost in performance from the VZ and was also used in my LIVENUP graphics series.

AN IDEAL SOUND GENERATING ROUTINE.

In the previous articles, I have mentioned some of the shortcomings of the various methods of generating sound on the VZ. An ideal sound generating routine should:

-Have the on and off intervals independantly variable. This is the capability to vary the 'duty cycle' of the sound and alters the 'timbre'.
-Have the ability to continuously vary the tone to obtain sound effects rather than just 'notes'.
-Have the tone duration presettable, but not necessarily independantly.
-Be able to be called from BASIC (via the USR command) and have the parameters passed from BASIC. (by POKEing into a table.)
-Be independant of BASIC as it is too slow for audio-frequency handling. This was a limitation to the programs provided so far.
-Be relocatable without the need for re-assembly so that it can be interfaced with other programs.

This looks like a pretty formidable programming specification! The first three criteria are achieved by providing suitable parameters to the routine. There are five in all:
-ON INTERVAL
-OFF INTERVAL
-ON INCREMENT
-OFF INCREMENT
-TONE DURATION
These 'specify' the sound envelope to be fed to the I/O latch. In essence, they are really timing loop constants for the machine language program.

The last three criteria are dealt with by suitable program design and implementation. Clearly the routine must be written in machine code. I have chosen to enter it as a series of DATA statements (in psuedo-assembler form) from the BASIC program. It could equally well be assembled using the ED/ASM. Unlike the other programs provided in this series, only one call from BASIC per sound, can be tolerated, so that good audio is obtained. This removes the annoying 'clicking' that occurs whenever the program passes backwards and forwards between BASIC and machine code. The requirement for re-locatability necessitates a few extra programming tricks so that parameters can be passed from the BASIC program to the machine code. This is taken care of in the program however by the use of temporary pointers stored in two-dimensional array VA%. As is my usual practice, I place machine code in a protected top-of-memory position - a procedure that I am sure most of you are familiar with.

Another useful feature that I have incorporated into the program, is the ability to record the sound effects on the VZeds cassette recorder. Remember in Part I, I mentioned that the I/O latch also controls the cassette port? To use this facility, just record on the cassette as the sounds are being created. They can be played back on any suitable 'Ghetto-Blaster'!

TONEGEN PROGRAM.

The program is quite easy to enter, but double check that lines 1010 to 1320 are correct. To gain a better understanding as to how the re-locatability and passing of parameters works, a debug dump is provided by removing the REMark from line 670. A perusal of the resulting print-out should clarify the method used. The parameters are directly POKEd into a 10 byte table reserved above the machine code held in top-of-memory. The program occupies 60 bytes and the table occupies 10 bytes of the 99 bytes reserved for use. The BASIC portion of the program is well commented and should be familiar to most. All of the REMarks can be omitted if you do not wish to enter them.

The 'working' section of the program is the machine code and the psuedo-assembler provided (lines 1010 to 1320) outlines the manner in which the sound effect is

obtained. The use of the alternate (or "back") register set on the Z80 is unusual and may confuse at first. I used this programming technique so that all of the parameters are held in on-chip registers and the program really zips along.

The program structure is really quite simple. There is an "outside" timing loop (T1) controlling the overall TONE DURATION. (lines 1050 to 1280) The loop counter for this is the alternate HL' register. This register is initialized in line 1040 and the use of this alternate register requires the frequent use of the EXX (exchange) instruction. (lines 1030, 1050, 1240 and 1290). There are two "inner" timing loops (T2 and T3) located within the "outer" one. The first of these controls the ON INTERVAL and the second one controls the OFF INTERVAL. Register HL (main register set) is the loop counter for each of these loops and is initialized in lines 1070 and 1160 respectively. For each of the timing loops familiar code for decrementing and testing for zero is provided in lines 1250 to 1280, 1110 to 1140 and 1200 to 1230.

The other two parameters are ON INCREMENT and OFF INCREMENT and are loaded into the BC and DE registers in lines 1010 and 1020. They are unmodified during the T1 loop. These values are summed with the HL register value to modify the duration of the T2 and T3 loops in lines 1030 and 1170. The incremented values are placed back in the table in lines 1090 and 1180. In this manner the sound is continuously variable.

The only remaining section of the code to discuss is the actual switching of the I/O latch located at 6800H. A suitable value is placed in the A register to switch the appropriate PAIRS of BITS to control both the piezo speaker and the cassette port. (recall Part I of this series.) Remember also that the pairs of bits must be complementary - i.e. one on and the other off. In line 1060 the A register is set to a decimal value of 36 during the ON INTERVAL in loop T2. This corresponds to the speaker pair - bits 0 and 5 - being set to off/on respectively and the cassette pair - bits 1 and 2 - being set to off/on. In line 1150, the A register is changed to a decimal value of 3 during the OFF INTERVAL in loop T3. This corresponds to an inversion of the 2 pairs of bits to on/off. In lines 1100 and 1190 the value in the A register is written to the I/O latch address and a resulting click is heard from the speaker or recorded on the tape. This sequence is repeated according to the "outer" timing loop T1 but with loops T2 and T3 being modified according to the ON and OFF INCREMENT parameters. In line 1300 and 1310 the A register is zeroed and written to the latch for an orderly return to BASIC.

As I said, it is a pretty easy program to follow and is a good preliminary introduction to assembly language techniques - particularly interfacing with the speaker and cassette hardware.

SOME PARAMETERS TO TRY.

It is fun to play around with the TONEGEN program and experiment with the effect each parameter has upon the resulting sound. Try the following:

100,100,0,0,1000 a square wave.
500,500,0,0,500 lower pitched square wave- longer loops.
10,10,0,0,5000 higher square wave.
10,0,0,0,10 10 clicks with varied duty cycle.
100,100,0,0,500 buzz.
200,20,0,0,1000
20,20,10,10,200
20,20,-32760,-32760,200
1000,1,-1,1,1000
5000,1,-10,10,500
1,1,1,1,1000
1,1,1,0,1000

and so on..... Try an visualize the values being placed into the HL, HL', BC and DE registers so that they can be related to the sound produced.

One final thing with this program. As you experiment with it, you will no doubt put the VZ into some very long loops that cannot be broken out of with the BREAK key. A useful addition to the machine language program is to have it look for the BREAK key in each of the loops. Have a go at making the necessary alterations. Another method is to use the Interrupt Vector to intercept a key sequence to return to BASIC and regain control.

We have investigated as far is possible in this series of articles, the software and hardware aspects of sound generation on the VZ. I trust that you will use some of the techniques to enhance your programming efforts. I hope that Z80 Assembler is becoming a little less mystic also.

```
1 '                    *************************
2 '                    ***   TONE GENERATOR   ***
3 '                    ***    BY R.B.KITCH    ***
4 '                    ***      17/4/87       ***
5 '                    ***   REF: VDU 3/16    ***
6 '                    *************************
9 '
10 CLS:PRINT@8,"TONE GENERATOR":PRINT@197,"LOADING MACHINE COD
E"
11 GOTO 110
20 '***CONVERT UNSIGNED TO SIGNED DECIMAL - PASSED IN UD & SD%
.
30 IF UD>32767 THEN SD%=INT(UD-65536) ELSE SD%=INT(UD)
35 RETURN
40 '***CONVERT UNSIGNED DECIMAL TO MSB & LSB - IN UD,MS%,LS%.
50 MS%=INT(UD/256):LS%=INT(UD-256*MS%):RETURN
60 '***CONVERT SIGNED TO UNSIGNED DECIMAL - PASSED IN SD% & UD
.
70 IF SD%<0 THEN UD=SD%+65536 ELSE UD=SD%
80 RETURN
100 '***LOWER TOM TO SAVE ROUTINE.
110 TM=256*PEEK(30898)+PEEK(30897)-100:'***RESERVE 99 BYTES.
120 UD=TM:GOSUB 50:POKE 30898,MS%:POKE 30897,LS%
130 CLEAR 100
                                       :'***RESETS ALL VARIABLE
S.
140 UD=0:TM=0:AD=0:SD%=0:MS%=0:LS%=0  :'***INITIALIZE STORAGE.
150 I%=0:DV%=0
160 DIM VA%(2,23),PA%(4),MS$(4)       :'***INITIALIZE ARRAYS.
170 TM=256*PEEK(30898)+PEEK(30897)    :'***TOM.
199 '
200 '***READ IN SOUND ROUTINE.
210 FOR AD=TM+1 TO TM+60
                                      :'***LOAD 60 BYTES.
220 UD=AD:GOSUB 30:READ DV%:POKE SD%,DV%
230 NEXT AD
240 UD=TM+1:GOSUB 50
                                      :'***SET USR POINTERS.
250 POKE 30863,MS%:POKE 30862,LS%
299 '
300 '***INITIALIZE POINTERS IN VA%().
310 FOR I%=0 TO 23
320 READ VA%(0,I%):UD=TM+VA%(0,I%):GOSUB 30:GOSUB 50
330 VA%(0,I%)=SD%:VA%(1,I%)=LS%:VA%(2,I%)=MS%
340 NEXT I%
399 '
400 '***POKE STORAGE LOCATIONS INTO ROUTINE.
410 POKE VA%(0,10),VA%(1,4):      POKE VA%(0,11),VA%(2,4)
420 POKE VA%(0,12),VA%(1,6):      POKE VA%(0,13),VA%(2,6)
430 POKE VA%(0,14),VA%(1,8):      POKE VA%(0,15),VA%(2,8)
440 POKE VA%(0,16),VA%(1,0):      POKE VA%(0,17),VA%(2,0)
450 POKE VA%(0,18),VA%(1,0):      POKE VA%(0,19),VA%(2,0)
460 POKE VA%(0,20),VA%(1,2):      POKE VA%(0,21),VA%(2,2)
470 POKE VA%(0,22),VA%(1,2):      POKE VA%(0,23),VA%(2,2)
499 '
500 '***LOAD SCREEN MESSAGES.
510 MS$(0)="RANGE OF+/-32767"+CHR$(10)+CHR$(13)+"   ON INTERVA
L"
520 MS$(1)="  OFF INTERVAL"
530 MS$(2)="  ON INCREMENT"
540 MS$(3)=" OFF INCREMENT"
550 MS$(4)="   TONE LENGTH"
599 '
600 '***FIND OUT PARAMETERS.
610 CLS:PRINT@8,"TONE GENERATOR":PRINT
620 FOR I%=0 TO 4
630 PRINT MS$(I%);:INPUT PA%(I%):SD%=PA%(I%)
640 GOSUB 70:GOSUB 50
650 POKE VA%(0,2*I%),LS%:          POKE VA%(0,2*I%+1),MS%
660 NEXT I%
670 'GOTO 20000     :'***DEBUG JUMP.
699 '
700 '***RUN TONE ROUTINE.
710 AD=USR(0)
799 '
800 '***GO AGAIN?
810 PRINT:INPUT"ANOTHER TONE (Y/N)";A$:IF A$="Y" THEN GOTO 610
899 '
900 '***RESET TOM ON EXIT.
910 TM=TM+100
                                       :'***ORIGINAL TOM.
920 UD=TM:GOSUB 50:POKE30898,MS%:POKE30897,LS%
930 CLEAR 50
999 '
1000 '***SOUND ROUTINE - VALUES OF 0 ARE RESET LATER.
1001 '***ALSO SETS CASSETTE PORT ON BITS 1&2.
1002 '***REGISTERS USED AF, BC, DE, HL & HL'.
1003 '***LENGTH 60 BYTES. NO STACK USED.
1010 DATA 237,75,0,0  :'    LD BC,(ONINC)
1020 DATA 237,91,0,0  :'    LD DE,(OFINC)
1030 DATA 217         :'    EXX
1040 DATA 42,0,0      :'    LD HL'(TONDUR)
1050 DATA 217         :'T1  EXX
1060 DATA 62,36       :'    LD A,00100100B
1070 DATA 42,0,0      :'    LD HL,(ONDUR)
1080 DATA 9           :'    ADD HL,BC
1090 DATA 34,0,0      :'    LD (ONDUR),HL
1100 DATA 50,00,104   :'    LD (6800),A
1110 DATA 43          :'T2  DEC HL
1120 DATA 125         :'    LD A,L
1130 DATA 180         :'    OR H
1140 DATA 32,251      :'    JR NZ, T2
1150 DATA 62,3        :'    LD A,00000011B
1160 DATA 42,0,0      :'    LD HL,(OFDUR)
1170 DATA 25          :'    ADD HL,DE
1180 DATA 34,0,0      :'    LD (OFDUR),HL
1190 DATA 50,00,104   :'    LD (6800),A
1200 DATA 43          :'T3  DEC HL
1210 DATA 125         :'    LD A,L
1220 DATA 180         :'    OR H
1230 DATA 32,251      :'    JR NZ, T3
1240 DATA 217         :'    EXX
1250 DATA 43          :'    DEC HL'
1260 DATA 125         :'    LD A,L
1270 DATA 180         :'    OR H
1280 DATA 32,215      :'    JR NZ,T1
1290 DATA 217         :'    EXX
1300 DATA 62,00       :'    LD A,00000000B
1310 DATA 50,00,104   :'    LD(6800),A
1320 DATA 201         :'    RET
1699 '
```

Continued on page 5

M/L ROUTINES USED IN ADVENTURE GAMES.

By David Wood

NOTE: If you are a beginner to Assembly Language, don't worry if you can't understand how these routines work. However what you should know is what they do and how to use them.

RESTORE LINE NUMBER

Those users who own other computers, or have tried to convert BASIC programs from other computers for the VZ may have noticed an extension for the RESTORE command. On the VZ this command causes the next piece of data to be read from the first item in the first DATA statement. Other computers can also use it in the form RESTORE 5030 (or any other line number, provided it exists) so that the next piece of data read will be the first item in line 5030 (or whatever). The VZ does not support this application of the RESTORE command. However a small Machine Language routine will simulate this. Recently I discovered that the routine had been placed in the wrong place, meaning that it didn't work and I had to alter it. Here is the correct version.

```
12 DATA237,91,33,121,205,44,27, 210,217,30,11,237,67,255,120, 201
13 FOR I=31273 TO 31288: READ A: POKE I,A: NEXT
```

```
ED 5B 21 79 001       LD    DE,(7921H)
CD 2C 1B    002       CALL   1B2CH
D2 D9 1E    003       JP NC, 1ED9H
0B          004       DEC BC
ED 43 FF 78 005       LD (78FFH), BC
C9          006       RET
```

The first line obtains the value in memory location 7921H (31009/10D) which contains the value of L sent in the BASIC command X=USR(L). This value is placed into the DE register. A CALL to 1B2CH is then carried out to find out where the line to be RESTOREd to occurs in memory and places that address in the BC register. If the line doesn't exist at all, a jump is made to the ?UNDEF'D STATEMENT ERROR routine and control is then returned to BASIC. If the line does exist, the BC register is decremented by one, and this value is loaded into location 78FFH (30974/5 decimal) which contains the location of the last piece of data read in the program. Because the memory location before the start of the line number to be RESTOREd to is loaded into this, the processor is "tricked" into thinking that last piece of data read was in the line before. Therefore the next piece of data will be read from the desired line.

This is useful for when you want to display a particular room description. These are stored in data statements which in the sample program start at line 5000, with each new description spaced by ten line numbers (IE room 1 at line 5000, room 2 at 5010, etc). If you wanted a particular description, you would need to RESTORE every time, and then read through all the data until you found the right one. This takes considerable amounts of time, and would be difficult to program as in the sample program, different descriptions may take up different numbers of data statements. By using the above routine, you can calculate the line number of the next piece of data you want read.

EG

```
L=4990+(R*10):GOSUB 5300

5300 POKE 30862,41: POKE 30863,122
5320 X=USR(L): POKE 30862,82: POKE 30863,121: RETURN
```

The USR command causes the computer to start executing machine code from the address pointed to by 30862/3, until it encounters the assembly language RET command where it will return to BASIC. At the the start of the subroutine at 5300 this pointer is altered to the start of the routine at 31273. When it has finished, the pointer is changed back to 31058 which is the start of the enhanced sound routine (below).

SOUND ROUTINE

This routine alters the VZ's sound capacity from 31 frequencies (highness or lowness) and 9 durations to 65535 frequencies and 65535 durations. While this may sound rosy, not all of these are very useful. Durations approaching 65535 are far too long to be of much use, and after the first few thousand frequencies, the rest are so slow they are only a series of clicks. This is still a huge improvement on the sounds used from BASIC. Unlike BASIC sounds, the highest frequencies are the low numbers. For instance, frequency one is so high that it is almost inaudible. This routine is very similar to the one that appeared in Bob Kitch's sound article a few months back, but was obtained from a different source.

```
15 DATA 243,1,100,0,33,20,0, 205,92,52,201
17 FOR I=31058 TO 31068: READ A: POKE I,A:NEXT
19 POKE 30862,82: POKE 30863,121
```

```
E3          001    DI
01 64 00    002    LD    BC,100D
21 14 00    003    LD    HL,20D
CD 5C 34    004    CALL  345CH
C9          005    RET
```

The values in lines 2 and 3 are dummy values, and must be POKEd into the right part of the routine from the BASIC program. The BC register contains the duration, and the HL register the frequency. A CALL is made to 345CH where the sound is executed, before the flow returns to BASIC. If you restrict both your frequency and duration to 255 or below, a sound can be executed by POKE 31063,freq: POKE 31060, duration: X=USR(0). If you want to use values outside this range you will need to use:

```
POKE 31064, INT(freq/256): POKE 31063, freq - (PEEK (31064)*256)
POKE 31061, INT(dur./256): POKE 31060, dur.  - (PEEK (31061)*256)
X=USR(0)
```

A wide range of sound effects can be created using this routine. Unfortunately the best way to discover them is by trial and error. Most of the best effects can be obtained using loops. Here are a few examples from the demo program and from Merkfruit Lodge.

DOOR OPENING

```
FOR I=110 TO 60 STEP -1: POKE 31060,1: POKE 31063,I: X=USR (0)
FOR J=1 TO I: NEXTJ,I
```

CRICKETS CHIRPING

```
FORJ=1TO40
FORI=1TO6: POKE 31060,40: POKE 31063,20: X=USR(0):NEXT
FORD=1TO100:NEXT:NEXT
```

(The above sound effect has limited use but it is probably the most realistic one I have come up with so far.)

CATS MEOWING

```
FORJ=1TO5:X=50-RND(20)
FORI=20+X TO50+X:POKE31063,I: POKE31060,4:Y=USR(0):NEXT
FORI=50+X TO40+X STEP-1: POKE 31063,I:y=USR(0):NEXT
FORD=1TO50+RND(300):NEXTD,J
```

AXE BLOWS

```
POKE 31064,5
FORI=1TO10:POKE31063,255: POKE 31060,2:X=USR(0): SOUND0,4: NEXT
POKE31064,0
```

(Note the use of the SOUND command in the above routine. This can often be used instead of a delay loop.)

BUBBLING TEST TUBES

```
POKE31060,40: FORJ=1TO50: ST= RND(20): FORI=ST TO ST+20 STEP 3
POKE31063,I:X=USR(0):NEXTI,J
```

MOSQUITOES

```
POKE31060,1:POKE31063,5: FOR I=1 TO 1000:X=USR(0):NEXT
POKE31063,2:FORI=1TO60: X=USR(0):NEXT
```

REMOTE-CONTROLLED CAR

```
POKE31064,1:POKE31060,1
FORI=1TO300:POKE31063, RND(50) +100:X=USR(0):NEXT: POKE 31064,0
```

GENERAL PROMPT

```
FORI=1TO3:POKE31060,40:POKE 31063,20:X=USR(0):NEXT
FORD=1TO100:NEXT
FORI=1TO3:POKE31063,40: X=USR(0):NEXT
```

With a little imagination and trial and error, you may be able to think of several other sound effects. Some other possibilities may include alarms or sirens, dripping taps, ticking clocks, motor vehicles such as cars or aeroplanes, assorted zaps and explosions, etc. If however, you wish to include tunes in your program, you may be better off using the SOUND command. This command was specifically designed for music programming, with each value of frequency (from 1 to 31) representing a note from A in the second octave to D in the fifth octave. (Note 16 is middle C.) The table of these values is in chapter eighteen of the BASIC REFERENCE MANUAL. Also a problem with the machine language sounds is that "durations" of the same value last longer for the higher notes than the lower notes. For example, a duration of 65535 lasts only about twenty seconds for frequency seven (a painfully high pitched noise), whilst the same value lasts for several minutes with a frequency of 65535 (occassional clicks). Although this is a rather exaggerated example this can be quite noticeable, even for sounds which are quite close together. This small loop may illustrate the point.

```
POKE31060,40: FORI=1TO255: POKE 31063,I: X=USR(0): NEXT
```

SIMULATING THE EXTENDED BASIC "DEFINT" COMMAND

In extended basics, the command DEFINT A-Z defines all variables A to Z as integer variables, without them having to be listed as A%,N%,Q%,Z% etc. Defining them as integer variables (variables which may only contain whole numbers between -32768 and 32767) causes the varibles to take up less memory in the simple variables table, and enables the program to operate more quickly.

The Variable Type Declartion Table exists from 30977 to 31002, having one location for each letter. The value POKEd into each location defines the variable type of any variable starting with this letter. The values are: 2 = integer, 3 = string, 4 = single precision, 8 = double precision. For example if location 30977 contained a two, the variables, A, AB, A9 and ANSWER would all be defined as integer

variables. However this can be overridden by the program - by using A$, A% or whatever in your program. Note that the default value for each location of the table is four, so all variables are single precision unless otherwise defined. To simulate the DEFINT A-Z command is quite simple, using a loop to poke the appropriate value into each of twenty-six locations.

```
14 FOR I=30977 TO 31002: POKE I,2: NEXT
```

FINDING THE LOCATION OF A PARTICULAR LINE

You may recall in the article I wrote about tape saving the adventure data, that I suggested it would be possible to alter the filename of the data file by POKEing the filename into the appropriate parts of the program. The major snag with this was that it would be difficult to find the exact location in the program where the CSAVE, CRUN or disk commands occurred. Fortunately, there is a useful machine langauge routine in ROM which will do this for us - the CALL to 1B2CH (also used in the Restore Line Number routine) will find the location of any line number in a BASIC program, provided it exists. The machine language routine used called from BASIC is therefore very similar to the Restore Line Number routine

```
ED 5B 21 79 001     LD    DE,(7921H)
CD 2C 1B    002     CALL   1B2CH
D2 D9 1E    003     JP NC, 1ED9H
ED 43 21 79 004     LD (7921H), BC
C9          005     RET
```

The routine is called from BASIC by the command LO=USR (line number)+5 The assembly language routine places the location of the start of the desired line in the variable LO. Five must be added as the first four locations of every line contain the memory location of the start of the NEXT line and the actual line number of the line. The next location in the line you wish to alter may contains the CSAVE, PRINT (or whatever) command in a token form. The sixth location contains a quotation mark (") but this will not be altered because the loop will automatically add at least one. The seventh and following locations contain the text in ASCII form which will be altered by the program. If you try to find a line that doesn't exist, the UNDEF'D STATEMENT ERROR routine is called. Here is the sample program:

```
10 DATA 237,91,33,121,205,44, 27,210,217,30,237,67,33,121, 201
20 FOR I=31273TO31287: READA: POKEI,A:NEXT
30 POKE30862,41: POKE30863,122
40 INPUT"INPUT YOUR FILENAME (16 CHRS)";A$
50 IFA$=""THEN STOP
60 A$=A$+"                ": A$= LEFT$(A$,16): '16 SPACES
70 LO=USR(110)+5: IF LO>32767 THEN LO=LO-65536
80FORI=1TO16
90Z=ASC(MID$(A$,I,1): POKE LO+I,Z
100 NEXT
110 PRINT"SAMPLE FILENAME."
120 GOTO 40
```

In line 70, note that the location must be converted into signed integer form.

ADVENTURE GAME WRITING - INITIALISATION

Most of this is placed at the end of the program, for reasons which were explained in an earlier edition. However the first part is left at the start of the program.

```
19 EL=31:NO=29:NV=27:G=17: X=USR(0):X=USR(0)
20 GOSUB 4000
```

This line defines a few constants which are used many times in the program. EL is the width of the screen, used for the automatic wrap-around subroutine (which ensures that a word isn't split between two lines on the screen). NO is the number of objects - both gettable and fixed objects - basically any noun that the computer knows. NV is similarly the number of verbs that the computer knows. G is the number of objects which may be picked up by the player. All of these (with the exception of EL) may need to be altered by the programmer during the course of the writing of the program as various elements of plot are added or taken away. This simply allows the programmer to fix everything up in one line instead of looking through the program for the many occurrences of these constants. (Ignore the X=USR(0): X=USR(0) in the line - this simply calls the beep routine twice, because when the LOAD routine restarts the program, the player needs to know to stop the tape.)

```
4000 DIMC(G),F(40),Z$(13),W(G)
4010 L=4900:GOSUB5300:FOR I=1 TO 13:READZ$(I):NEXT
4020 X$= "N??S??W??E??HELINVGETEXALOODRO LEAOPEREAUNLLIGEXTDO?"
4030 X$=X$+ "WINDIGGIVWEAKICEATSTAPLASAVLOA "
4040 Y$= "LISMAPTAPBOOFOOSTIGASMOUCAREAR BONKEYBATTORROCFRIWASDOG"
4045 Y$=Y$+ "CATFOLSTEBOOBROHOMBUSDOODOOSOC DRA"
4046 'LEAVE OUT THE SPACES IN THE ABOVE LINES
4050 FORI=1TOG:READC(I):NEXT: FORI=1TO9:READJ:F(J)=1:NEXT
4055 F(25)=10:F(22)=9
4060FORI=1TOG:READW(I):NEXT
4065 R=25:S=120:R$="OK... GOOD LUCK"
4070 RETURN
```

Line 4000 dimensions arrays for some of the data from the program. Array C sets aside G (which equals 17) spaces for information on where each object is, and whether the player is carrying it or not. Array F sets aside 40 spaces which determine whether a gettable object is "invisible" or not, and other miscellaneous information such as whether a certain door is open, or whether the torch is lit or not. The Z$ array simply carries some useful words for the start of room descriptions, and descriptions of objects. In the demo program the words are "in ","on ","by ","at ","a "," an ","","some ","the ","north", "south","west" and "east". Note the spaces which accompany some of the words. Finally the W array carries the relative weights of the gettable objects. The player's strength, and the sum of the weights of the objects he/she is trying to carry determines whether a certain object can be carried or not.

Lines 4020 and 4030 put the first three letters of every verb into the string X$. Where the verb contains less than three letters, use question marks to "fill up" the remaining spaces. EG X$="N??S??W??E??HELINV..... Similarly 4040 and 4045 put the first

three letters of every noun into Y$.

Line 4050 reads the room location number of every gettable object into the C array. If the player picks up the object the location number is replaced by a zero. Another loop reads the object numbers of the objects which are deemed "invisible" (that is, they cannot be immediately seen by the player, if hidden in a cupboard for example). For these a one is placed in the array and for visible objects a zero. Object 14 is invisible, so F(14)=1, but object 15 isn't so F(15) remains equal to 0. Line 4055 puts other values in the array - that there are ten servings of food left, and that the dog is in room nine.

The weight values are read into the W array with light objects such as a key (1) to heavy objects such as a fridge (50).

The initial Room that the player occupies in this case is set to 25, and the Strength to 120. R$, which is the message displayed along with the room description is set as "OK... GOOD LUCK".

At line 5000 in the demo program, the DATA statements for the room descriptions start. To save memory, the words "YOU ARE " are left off, and added to the descriptions later in the program. Following is some code for the next two words in the descriptions. EG.

5000 DATA"13YOUR BROTHER'S BEDROOM.....

OR

5010 DATA"15NORTHERN PART OF THE LOUNGE.....

This is because room descriptions start with only a few different words.

FIRST WORD

1 IN
2 ON
3 BY
4 AT

You may also like to use others like NEAR or FACING or no word at all for where your description doesn't begin with any of these words.

SECOND WORD

1 A
2 AN
3 (nothing)
4 SOME
5 THE

You may need to use several lines for each room description to overcome the problem that the VZ only allows 64 characters per line. But always make sure that each new room description is the same number of line numbers apart, so you can make an equation for the RESTORE LINE NUMBER routine to be able to read to correct description: eg L=4990+(R*10)

At the end of each description you place your movement codes. In this way they also act as end of description markers. eg

5004 DATA " ARE IN HERE. 1011"

OR

5012 DATA "A BOOKSHELF. 1010"

(Remember that the movement codes indicate which directions the player can move. In the first instance, south is the only option and in the second the player can move south or east, but not north or west.)

At line 4700 in the demonstration program are the descriptions of gettable objects. At the start of each description is a code based on the second word of the room description codes (ie. A ,AN ,"" (not used in this case), SOME , and THE (also not used)). Otherwise these are fairly straightforward, placed five lines apart. eg

```
4770 DATA1LARGE AND VERY HEAVY ROCK
4775 DATA2OLD WHITE FRIDGE
4780 DATA1LARGE WASHING MACHINE
```

(Make sure you place these in the same order as you put the first three letters of the nouns into Y$. Also in Y$, use the first three letters of the nouns themselves (ie ROCFRIWAS) rather than the descriptions.)

The other data lines are even more straightforward. Line 4955 contains the initial location (room number) of each object (same order as they appear in Y$). Line 4960 contains the object numbers of the objects that are "invisible." (The object number is the order in which they appear in Y$ and the object descriptions - the first object is number one, the second is two, etc). The third contains the the relative weights of each object in order.

If you wish, you may like to type in your initialisation and data statements before you have finished planning your verb subroutines.

The HIGH Scores

GAME	SCORE	LEVEL	HOLDER
DAWN PATROL	52500		David Wood
CRASH	573		Peter Watson
DIG OUT	24400		Paul Frantz
HAMBURGER SAM	39500		Stephen Frantz
LADDER CHALLENGE	22530		Peter Watson
KAMIKAZE	16420		Peter Watson
TEN PIN BOWLS	206		Bernice O'Mahoney
VZ INVADERS	30160		Peter Watson
GALAXON	313260		Kenley McLean
PENGUIN	1350		Jason Oakley
LUNAR LANDER	3600		Jason Oakley
SUPER SNAKE	1918	Novice	Peter Watson
MAZE OF ARGON	73258		Peter Watson
ASTEROIDS	110000		Peter McLean
CIRCUS	1210		Peter Watson
PANIK	7700		Peter Watson
HOPPY	25550		Matthew McLean
GHOST HUNTER	23400		Chris McLean
STAR BLASTER	480 units left		Matthew McLean
KNIGHTS & DRAGONS	3700	Easy	Peter Watson

TRADING POST

EPROMS for EXTENDED DOS. and BASIC
Are available from
Bob Kitch
7 Eurella Str.
KENMORE Q'ld. 4069

FOR SALE

COMPUTMATE II. This is an electronic hand held Spelling checker, appointment file, address file, clock, calendar and calculator.
Price $70.
Ron Allen. 2 Orlando St. Hampton. Vic.3188. Ph. 598 4534.

####

For Sale.
V.Z 300, with casserre recorder and 25 tapes.
Fair offer.
Janice Hardy 16 Chevell St. Corowa 2646.Ph.060 332 980.

####

Wanted to BUY.
V.Z.200. In good order. Good price paid.
Ben Hobson. P.O.Box. 255 Quirindi. N.S.W. 2343.

####

OTHER V Z USER GROUPS

H.V.V.Z.U.G
P.O.Box 161
JESMOND NSW.2299.

DISKMAG
P.O.Box 600.
Taree NSW. 2430.

CENT.VIC.COMP.Club
24 Breen St.
BENDIGO VIC 3550

BRISBANE VZUG
63 Tingalpa St.
WYNUM West. Q'ld. 4178

Graeme Bywater
P.O.Box 388
MORLEY W.A. 6062